(19) Europäische Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 620 693 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94302220.2**

(51) Int. Cl.[5] : **H04Q 3/00**

(22) Date of filing : **28.03.94**

(30) Priority : **14.04.93 GB 9307647**

(43) Date of publication of application :
19.10.94 Bulletin 94/42

(84) Designated Contracting States :
DE ES FR IT SE

(71) Applicant : **GPT LIMITED**
New Century Park
P.O. Box 53
Coventry, CV3 1HJ (GB)

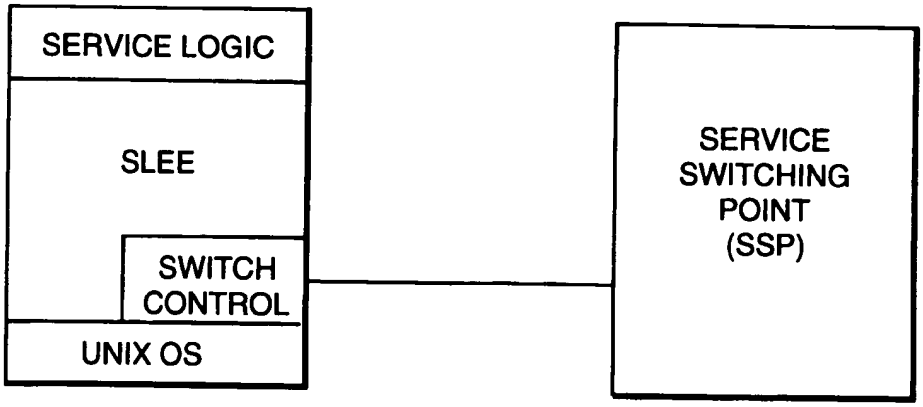
(72) Inventor : **Weiner, Steven Michael**
68 Manfield Avenue
Walsgrave, Coventry CV2 2HN (GB)
Inventor : **Tavanyar, Simon Edwin**
1387 Black Willow Trail
Altamonte Springs, Florida 32714 (US)
Inventor : **Sanders, Jeffrey Allen**
2301 Pebble Vale 511
Plano, Texas 75075 (US)

(74) Representative : **Branfield, Henry Anthony**
The General Electric Company, p.l.c.
GEC Patent Department
Waterhouse Lane
Chelmsford, Essex CM1 2QX (GB)

(54) Telecommunications switch control.

(57) A telecommunications switching system includes a Switch Control component for a Service Logic Execution Environment (SLEE), wherein the rest of the SLEE and the Service Logic Programs (SLPs) are running above the Switch Control Component and the Switch Control provides an interface resulting in the SLEE and SLPs being independent of a switch to which they are connected. The switch control contains 'mix and match' modules which each perform well-defined functions.

Fig.1.



EP 0 620 693 A2

The present invention relates to the Switch Control component of a Service Logic Execution Environment (SLEE), allowing the rest of the SLEE, and the Service Logic Programs (SLPs) running above it, to be independent of the switch to which they are connected. This is achieved by designing the switch control to contain 'mix and match' modules which each perform well-defined functions.

According to the present invention there is provided a telecommunications switching system including a Switch Control component for a Service Logic Execution Environment (SLEE), wherein the rest of the SLEE and the Service Logic Programs (SLPs) are running above the Switch Control Component and the Switch Control provides an interface whereby the SLEE and SLPs are independent of a switch to which they are connected.

The present invention will now be described, by way of example, with reference to the accompanying drawings, in which :-

Figure 1 shows a diagrammatic view of Switch Control within an Intelligent Network (IN) configuration;

Figure 2 shows a diagrammatic representation of Switch Control Configurations within SLEE;

Figure 3 is a block diagram illustrating the transmission of a message from a SLEE to a switch; and

Figure 4 is a block diagram illustrating the receipt of a message from a switch to the SLEE.

This forms part of an Advanced Intelligent Networks (AIN) Service Logic Execution Environment (SLEE) which itself forms a part of an Adjunct and Service Control Point (SCP) system. The SLEE developed from the requirements set out by Bellcore in the following publications:

1. FR-NWT-001132, AIN Release 1 Service Logic Program Framework Generic Requirements
2. TA-NWT-001124, AIN Release 1 SLEE Generic Requirements

The SLEE provides a higher-level layer of functionality above the Operating System (OS) protecting the service implementations above from changes in the OS, and providing them with more telecommunications oriented functionality than a general-purpose OS.

Services are implemented above the SLEE as a number of Service Logic Programs (SLPs) which make use of the Bellcore defined Application Programming Interface (API) presented by the SLEE.

An important part of the SLEE is the Switch Control function which handles the protocols and higher level messages used between the Service Switching Point (SSP or Switch) and the Service Control Point (SCP).

As typically a number of switches (SSPs) are supported with different SSP/SCP message sets, some standard and some proprietary, carried over different communication channels, it is important that the switch control function within the SLEE is such that a variety of message sets and communication channels be supported without severely impacting either the design of the switch control function itself, or the Service Logic Programs running on the SLEE. By achieving this, there is the ability to write Service Logic Programs which are independent of the SSP message set and communication channel.

The current requirement is for a SLEE, and therefore the switch control part of it also, to be designed to interface to a specific switch. The ability to give service developers independence from the particular target switch used, by designing the switch control function to hold a set of mix and match modules, is provided.

Switch Control forms the interface between the SLEE and the various Switches to which the SLEE is connected. This interface is a modularised interface allowing the selection of different message sets and different physical interfaces.

Figure 2 shows the typical configurations that will be produced in the different versions of the switch control.

The task of Switch Control is to convert messages between the block structure format used within the SLEE and the transfer syntax used for communicating with the switch.

Switch Control achieves this by calling a number of functions which performs a different portion of the required conversion. Block diagrams showing the processing performed by Switch Control in both directions are shown in Figure 3.

The functions performed by these different blocks are:
Block_to_Transfer and Transfer_to_Block.

The message for transmission to the Switch is in a block structured format, which may be a C structure that contains all the parameters that make up the message. Block_to_Transfer converts this to a TCAP parameter set by establishing what parameters are actually present in the message, calculating their length, if necessary, and creating the corresponding TCAP elements.

Transfer_to_Block performs the reverse process, by passing through a TCAP parameter set establishing which parameters are present in the message and populating the required part of the Block structure message.

Both Block_to_Transfer and Transfer_to_Block are message set dependent. Convert_to_TCAP and Convert_from_TCAP.

Convert_to_TCAP takes the TCAP parameter set generated by Block_to_Transfer and places it in a TCAP component portion which itself is added to a TCAP transaction portion. It sets all the necessary values within the Transaction and Component portions.

Conv rt_from_TCAP xtracts th TCAP parameter set from the TCAP Transaction and Compon nt portions, whilst correctly processing the data contained within the portions.

Conv rt_to_TCAP and Convert_from_TCAP are message set independ nt. They are however dep ndent on the v rsion/incarnation f TCAP us d. Diff rent variants of the two routin s would b us d f r diff r nt TCAP variants e.g. ANSI Issue 1, ANSI Issue 2, CCITT Blue Book, CCITT White Book.

Message_Encapsulation and Message_Decapsulation.

These two routines add/remove the necessary header information for communication with the Switch to which the SLEE is connected. They handle the reception/transmission of the messages communicating with any hardware device drivers that might be present.

Message_Encapsulation and Message_Decapsulation are both Message set and TCAP variant independent. They are however switch dependent.

The overall dependencies of the various components are:

	Message Set Dependent?	TCAP Variant Dependent?	Switch Dependent?
Block_to_Transfer Transfer_to_Block	YES	NO	NO
Convert_to_TCAP Convert_from_TCAP	NO	YES	NO
Message_Encapsulation Message_Decapsulation	NO	NO	YES

This structure for Switch Control allows for the changing of any one of the components without affecting the any of the others. For instance to change a Switch Control that was using the CS-1 message set with Blue Book TCAP from a System-X implementation to an EWSD implementation the only area that would require changing would be the Message_Encapsulation/Message_Decapsulation.

Glossary of Abbreviations and Terms

AIN	Advanced Intelligent Networks
ANSI	American National Standards Institution
API	Application Programming Interface
CCITT	Consultative Committee on International Telegraphy and Telecommunications
CCS.1	Connection Control Socket, part 1. A SCP/SSP interface designed by GPT for use with the System-X switch
CS-1	Capability Set 1. CCITT/SCP/SSP standard interface
DCO	Digital Central Office (DCO Switch)
EWSD	Switch used in equipment supplied by Siemens AG
OS	Operating System
SCP	Service Control Point
SLP	Service Logic Program
SLEE	Service Logic Execution Environment
SSP	Service Switching Point
System-X	Digital switch used in equipment supplied by GPT Limited
TCAP	Transaction Capabilities Application Part
UNIX	A Computer Operating system. UNIX is a trademark of USL Inc.

Claim

1. A telecommunications switching system including a Switch Control component for a Service Logic Execution Environm nt (SLEE), wh rein the rest of the SLEE and the Service Logic Programs (SLPs) are running above the Switch Control Component and the Switch Control provides an interface wh reby th

SLEE and SLPs are independent of a switch to which they are connected.

2. A Switch Control as claimed in Claim 1, wherein the SLEE provides a higher level layer of functionality above the Operating System (OS).

3. A Switch Control as claimed in Claim 1 or 2, which handles the protocols and higher level messages between a Service Switching Point (SSP) and a Service Control Point.

4. A Switch Control as claimed in Claim 1, 2 or 3, having means whereby messages are converted between the block structure format used within the SLEE and the transfer syntax for communicating with the switch.

Fig.1.

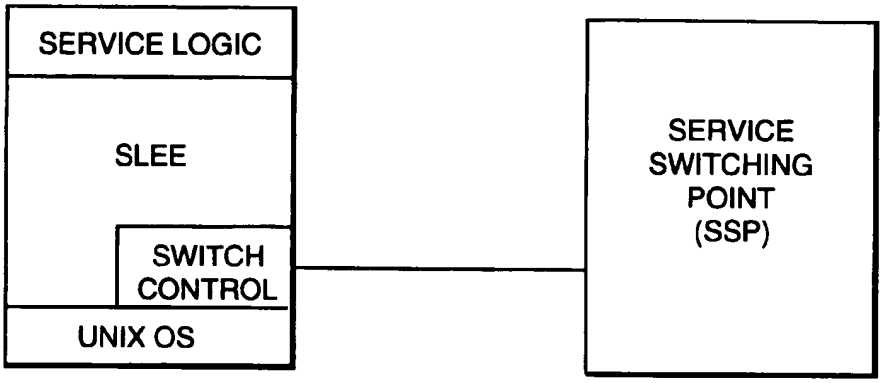

SERVICE LOGIC
SLEE
SWITCH CONTROL
UNIX OS
SERVICE SWITCHING POINT (SSP)

Fig.2.

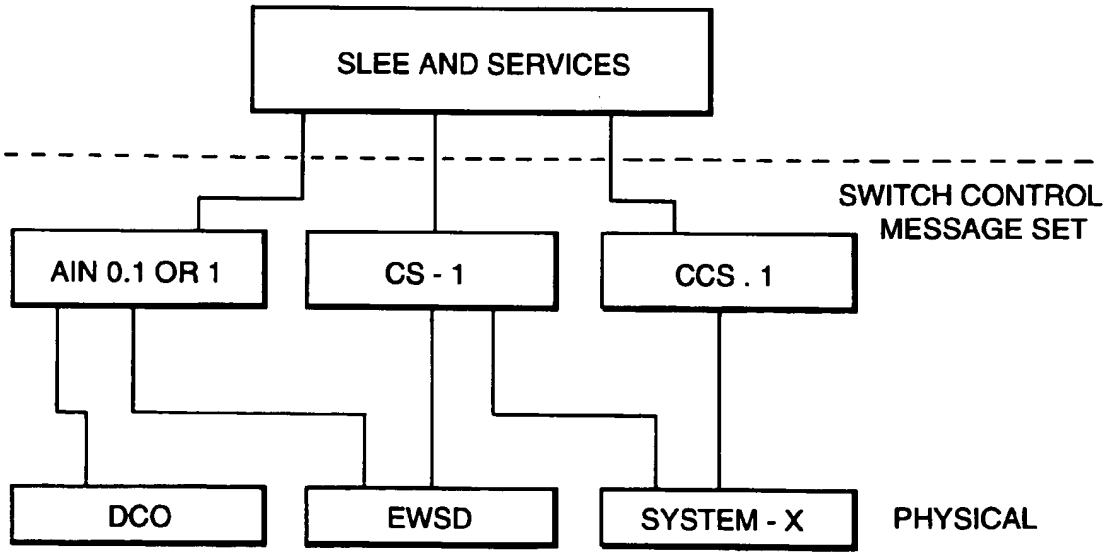

SLEE AND SERVICES
SWITCH CONTROL MESSAGE SET
AIN 0.1 OR 1
CS - 1
CCS . 1
DCO
EWSD
SYSTEM - X
PHYSICAL

Fig.3.

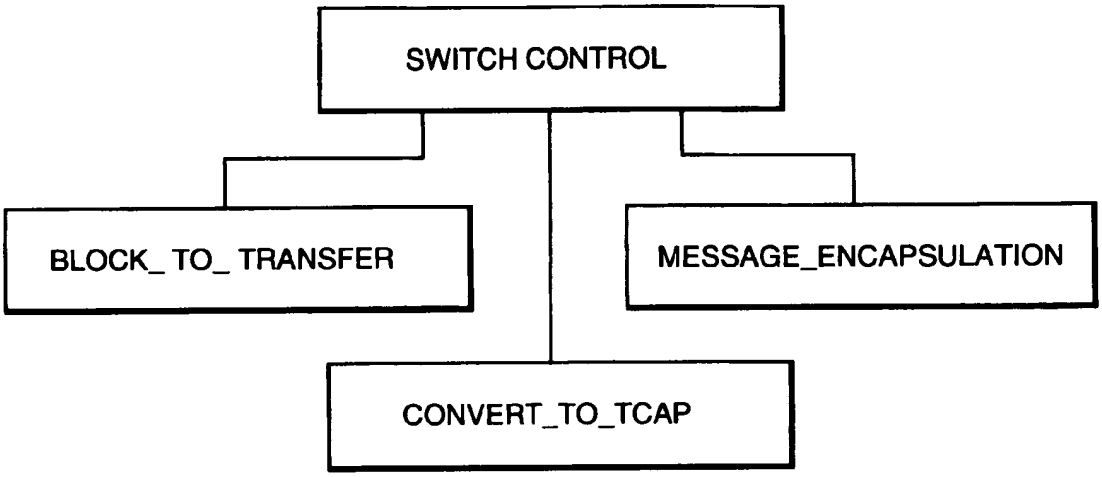
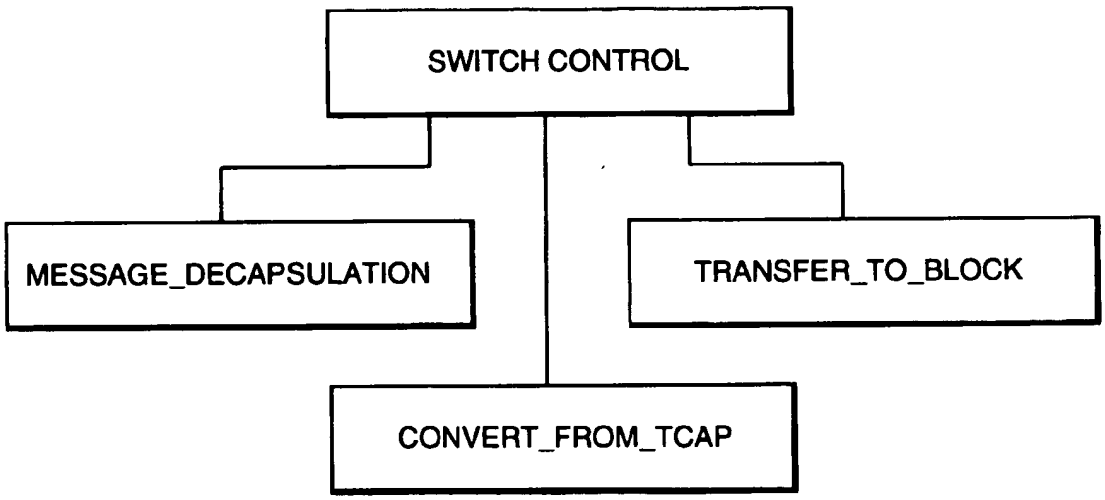

SWITCH CONTROL
BLOCK_ TO_ TRANSFER
MESSAGE_ENCAPSULATION
CONVERT_TO_TCAP

Fig.4.


SWITCH CONTROL
MESSAGE_DECAPSULATION
TRANSFER_TO_BLOCK
CONVERT_FROM_TCAP